DOCUMENT RESUME

ED 156 204 IR 006 437

AUTHOR Tucciarone, John P.
TITLE Using Networks to Net Results.
PUB DATE Jun 77
NOTE 8p.; Paper presented at the Conference on Computers in the Undergraduate Curricula (East Lansing, Michigan, June 20-22, 1977); For entire proceedings, see IR 006 142

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS *Algorithms; Computer Programs; *Computer Science Education; Course Descriptions; Critical Path Method; Higher Education; *Networks; *Programing; *Programing Problems; Undergraduate Study
IDENTIFIERS BASIC; Mercy College NY

ABSTRACT

As part of an introductory computer programming course, students were assigned seven programs selected to expose them to the varied applications of BASIC. Three additional programs were selected by the students themselves from other courses or from topics in the text which had not been covered in class. The topic which seemed to arouse the greatest interest was the Kemeny-Kurtz treatment of a critical path-network flow problem, a topic not normally treated in introductory courses. Even though the course, which had no prerequisites, attracted students with a varied degree of quantitative competence, most students were able to handle the programs effectively once the ideas were presented in a concrete framework and the algorithms were carefully explained. Three network problems selected on the basis of this experience are described: (1) minimum tree in a network; (2) minimal time path through a network; and (3) critical path through a network. Five student written programs discussed in the article are included as an appendix. (Author/CMV)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED156204

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

USING NETWORKS TO NET RESULTS

BY

John P. Tucciarone

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Ted Sjoersma

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

IR006437

Using Networks to Net Results

John P. Tucciarone
Mercy College
Dobbs Ferry, New York 10522
(914) 693-4500

Introduction

This paper is an outgrowth of a course in introductory computer programming given at Mercy College, Dobbs Ferry, New York, during the spring of 1976. There were two sections of the course, each taught by the author. The enrollment in both sections totaled fifty-one and consisted mainly of freshmen and sophomores whose ages ranged from eighteen or nineteen years on up to fifty-three.

The course was open to all students and it was devoted to instruction in the fundamentals of programming in BASIC, with students using remote teletype terminals to run programs. In addition to classroom lectures and "shop sessions" during which the author personally supervised the running of programs, the students were assigned a total of ten programs to complete in the course. These programs were selected to expose the students to the varied applications of BASIC. The programs included: looping exercises, character string manipulation, financial examples, matrix manipulation, and game simulation. Some of the discussions and assigned programs were selected directly from the text, *Basic Programming*, by John G. Kemeny and Thomas E. Kurtz.

In addition to the seven programs whose topics were described above, the students were encouraged to select the remaining three from applications in other courses or from the remaining topics in the text, which had not been treated in class lectures. Some class time was devoted to the highlighting of several of these topics.

The topic which seemed to arouse the greatest interest was the Kemeny-Kurtz treatment of a critical path-network flow problem. The majority of the students in the course were planning to take business-oriented courses as part of their undergraduate curriculum, and approximately one-third were planning to major in business. Perhaps that fact explains why such interest was generated in a problem which would normally appear to a layman to be difficult to solve, albeit timely and practical. The author used the opportunity afforded by the curiosity of the students to discuss several elementary network problems. The author discovered that after a brief treatment of algorithms and suggestions for their implementation, and the organization of data, most of the students who elected to work on the network problems were able to write efficient programs. This paper is devoted to that experience.

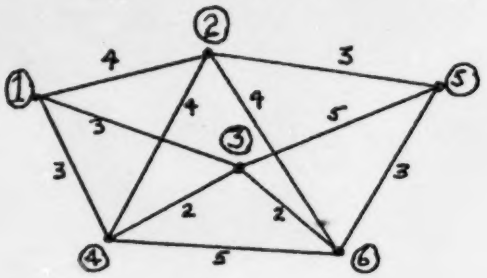
In the following, three network problems are treated. The selection of these was made with two criteria in mind. First, that the problems be practical and easy for a student to identify with. Second, that simple algorithms existed for the solution of each, so that a reasonably elementary program could be written to model the algorithm. With these criteria in mind, the author selected: (1) A Minimum Tree in a Network, (2) A Minimal Time Path Through a Network, and (3) A Critical Path Through a Network.

A Minimum Tree in a Network

Problem: Given a network of cities with interconnected highways, find the cheapest way to upgrade the existing roads so that each city is connected to every other by an upgraded route, and without the construction of new roads. That is, seek a tree without closed loops which includes all the intersections

(cities), and minimizes the sum of the distances represented by each road (or edge) in the network.

Since the students were totally unfamiliar with the methods of linear programming, a simple approach was suggested for the solution of the problem. All the edges in the network were arranged in increasing length. The shortest was then chosen first and to that was added the shortest of the remaining edges which had one node (or intersection) in the tree, but not both. The solution tree is then complete when all nodes are included in the tree. The following example was considered:



Network I

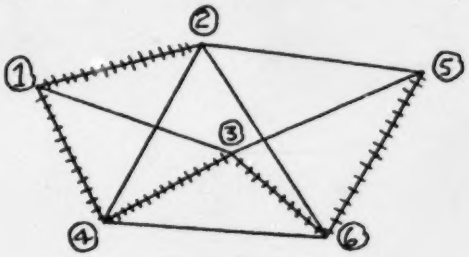
The numbers in circles indicate cities and the remaining numbers indicate the distances between them.

The edges were arranged in matrix form producing an 11 by 3 matrix in which the first and second columns represented the two nodes forming each edge and column 3 represented the length of that edge. The matrix is ordered on column 3, that is, the higher the row number in the matrix, the longer the edge. Otherwise, there is no priority as to which nodal end of an edge is listed in column 1 or in column 2.

Student written Program I (see Appendix) produced the following listing:

```
3  4  2
3  6  2
1  4  3
5  6  3
1  2  4
```

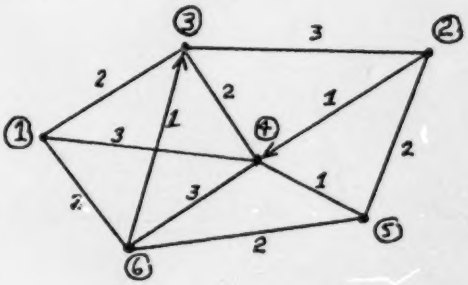
which is interpreted as meaning that the edges formed by the nodes in columns 1 and 2 form the minimal tree and the sum of the numbers in column 3 gives the total length of the tree, 14. The minimal tree is depicted as follows:



A Minimum Time Path in a Network

Problem: Given a grid of streets and intersections in a certain city with the time needed to traverse a given street (from one intersection to another) at a given time of day, find the shortest time path from one intersection to any other.

The minimization could well involve distance rather than time but for students already familiar with the traffic congestion of metropolitan New York City, the minimum time path seemed to be the more appealing choice. The following example was considered:



Network II

The numbers in circles represent intersections (randomly numbered), and the remaining numbers represent the time required to traverse that edge. All edges (or streets) are assumed to be two-way unless indicated one-way, such as 6-3, and 2-4.

We start by arranging the data in two matrices, A, the intersections, and B, the edges. Matrix B, a 20 by 3 matrix, contains

the 20 edges, counting two-way streets twice and one-way streets once. Column 1 contains the intersection number of each edge starting with 1, then 2, etc., in numerical sequence. Column 2 contains the time value for each, and column 3 contains 0 for each entry except when that entry marks the last edge having a given starting intersection number, in which case column 3 contains a 1. Thus, a search of column 3 in matrix B will yield groups of edges having the same starting intersection.

Matrix A, a 6 by 3 matrix, contains the 6 intersections. Column 1 contains the associated row number of matrix B which marks the start of the group of edges having a starting intersection equal to the row number of A. Column 2 contains the time values which are set at 0 for the intersection from which minimal time paths are sought. The remaining entries in Column 2 consist of time values which exceed the possible total for the entire network. In the example under consideration, 50 was selected as the starting value. Column 3 is set at the start with a value of 1, corresponding to the intersection from which the minimal time paths are sought, and 0's elsewhere.

For the example, matrices A and B are as follows:

A

1	0	1
4	50	0
7	50	0
10	50	0
14	50	0
17	50	0

B

3	2	0
6	2	0
4	3	1
3	3	0
4	1	0
5	2	1
1	2	0
4	2	0
2	3	1
5	1	0
6	3	0
3	2	0
1	3	1
4	1	0
2	2	0
6	2	1
1	2	0
3	1	0
4	3	0
5	2	1

The algorithm consists in starting at a given intersection and going from there to all those intersections which can be reached from it, calculating the time to traverse, and then to successive intersections. If an intersection is reached which has been reached before, a new travel time is substituted if it is smaller. When all the intersections have been processed, the algorithm terminates.

Student written Program II (see Appendix) produced the following listing:

0	0
5	0
2	0
3	0
4	0
2	0

which indicates that the shortest time path from intersection 1 to those numbered 2, 3, 4, 5, and 6, is given by the values in column 1. That is, the shortest time path from intersection 1 to intersection 2 is a total of 5 time units.

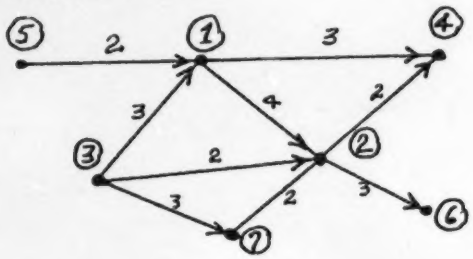
To find the actual route once the minimal time is known, the following procedure was used. We work backward from the intersection one desires to reach by going to the pre-intersection whose time value differs from the time value up to the current intersection by an amount equal to the time value for that street. Of course, this procedure should be modified to include those cases where the actual minimal time path is not unique. In our example, the minimal time from intersection 1 to 5 is 4 units, but there are two minimal paths, namely, 1-6-5 and 1-4-5.

Student written Program III (see Appendix) produced the following listing when the intersection 2 was input: 2-3-1. This indicates that the minimal time path for a time of 5 units from intersecton 1 to intersection 2 is the path 1-3-2.

A Critical Path Through a Network

Problem: Given a project with a fixed number of events to be completed and a process or activity which culminates in the completon of each event, several questions may be asked. In the directed network, directed since we assume there is some precedence in the

flow through the network, what is the shortest time within which the entire project may be completed? What are the earliest times or latest times each event may be completed without affecting the best time for completion of the entire project? The following example was considered:

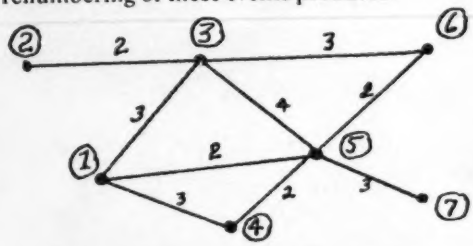


Network III

The events are numbered in circles and the directed edges indicate the intervening activity with the time for completion of each marked.

First we proceed to renumber the event circles so as to obtain a sequential topological ordering of the network. This process consists in the arranging of the data into two matrices, E and A. In the example, matrix E is a 7 by 2 matrix. Column 1 contains the row number of the activity matrix A in which the edges having a starting event number 1, 2, etc. begins. Column 2 contains the number of predecessors that each event has. Matrix A is a 9 by 2 matrix. Column 1 contains 0's everywhere except for 1's to indicate the final edge of a group whose starting event corresponds to that row number in E. Column 2 lists the successor event for the particular edge.

Student written Program IV (see Appendix) produced the following listing: 3-5-1-7-2-4-6, which indicates the correct topological ordering of the events in the network. A renumbering of those events produces:



We are now ready to find the earliest (latest) dates on which each event may be completed. We arrange the data in one 9 by 3 matrix, S. Columns 1 and 2 in S represent the nodal ends of each edge and column 1 is arranged according to the topological order determined above. Column 3 contains the lengths of each activity. The algorithm then proceeds by assigning a value of 0 to each of the events 1 and 2 since they have no predecessors in the network. We then process the list once in descending fashion, adding to the current value of an edge (given in column 3) the value of the second node (given in column 2). If the result is greater, we replace. After the list has been processed, we select as the value for each event or node, the largest value from column 3 which is associated with the occurrences of that node in column 2. In the example, matrix S has the form:

```
1 3 3
1 5 2
1 4 3
2 3 2
3 6 3
3 5 4
4 5 2
5 6 2
5 7 2
```

Student written Program V (see Appendix) produced the following listing:

```
EVENT 3 HAS VALUE 3
EVENT 4 HAS VALUE 3
EVENT 5 HAS VALUE 7
EVENT 6 HAS VALUE 9
EVENT 7 HAS VALUE 10
```

We interpret these values to mean that a total project time of 10 units are required and column 2 gives the earliest times when each event may be completed. A similar procedure may be used to calculate the latest times.

Comments

The author found that a topic which is not normally treated in an introductory course in programming turned out to be one which elicited active interest. The course had no prerequisites and thus attracted students

with a varied degree of quantitative competence. The author found, however, that most students were able to handle the programs effectively once the ideas were presented in a concrete framework and the algorithms were carefully explained.

The results raised two questions which prompted the author to write this paper. (1) Given the non-academic sophistication of many college students today, might a topic such as networks, usually considered advanced, be fruitfully studied at the introductory level? (2) For students who may be weak in arithmetic skills, might "on the job training" in the nature of actual problem-solving and program-writing improve those skills? Based on the experience of this course, the author is inclined to answer "yes" to each. Once the interest of a student is aroused and he or she feels the empirical need for a solution to a problem, a need which he or she can relate to, and understand, very little more in the way of motivation is needed from the instructor.

References

Kahn, A. B. "Topological Sorting of Large Networks." CACM. 1962. Vol. 5, pp. 558-562.

Klein, M. "Scheduling Project Networks." CACM. 1967. Vol. 10, pp. 225-230.

Knuth, D. E. Fundamental Algorithms, Reading, Mass., Addison-Wesley, 1968.

Nicholson, T.A.J. "Finding the Shortest Route Between 2 Points in a Network." *Computer Journal*, 1966. Vol. 9, pp. 275-280.

Appendix

Program I

```
010 DIM S(11,3), M(6), T(11)
020 MAT INPUT S
030 LET K=C=1
040 LET M(1)=S(K,1)
050 LET M(2)=S(K,2)
060 LET T(C)=K
070 LET L=2
080 LET K=K+1
090 IF K=12 THEN 290
100 FOR I=1 TO L
110 IF S(K,1) M(I) THEN 200
120 FOR J=1 TO L
130 IF S(K,2)=M(J) THEN 280
140 NEXT J
150 LET L=L+1
160 LET M(L)=S(K,2)
170 LET C=C+1
180 LET T(C)=K
190 GO TO 280
200 NEXT I
210 FOR I=1 TO L
220 IF S(K,2) M(I) THEN 270
230 LET L=L+1
240 LET M(L)=S(K,1)
250 LET C=C+1
260 LET T(C)=K
270 NEXT I
280 GO TO 80
290 FOR A=1 TO C
300 PRINT S(T(A),1);S(T(A),2);S(T(A),3)
310 NEXT A
320 END
```

Program II

```
010 DIM A(6,3), B(20,3)
020 MAT INPUT A, B
030 FOR K=1 TO 6
040 IF A(K,3) 1 THEN 150
050 LET X=A(K,2)
060 LET Y=A(K,1)
070 LET Z=X+B(Y,2)
080 LET D=B(Y,1)
090 IF Z A(D,2) THEN 120
100 LET A(D,2)=Z
110 LET A(D,3)=1
120 IF B(Y,3)=1 THEN 180
130 LET Y=Y+1
140 GO TO 70
150 NEXT K
160 MAT PRINT A
170 STOP
180 LET A(K,3)=0
190 GO TO 30
200 END
```

Program III

```
010 DIM A(6,3), B(20,3)
020 INPUT X
030 PRINT X;
040 IF X=1 THEN 150
050 LET Y=A(X,2)
060 LET Z=0
070 FOR K=1 TO 20
080 IF B(K,1) X THEN 130
090 LET N=Z+1
100 IF Y-A(N,2) B(K,2) THEN 130
110 LET X=N
120 GO TO 30
130 LET Z=Z+B(K,3)
140 NEXT K
150 END
```

Program IV

```
010 DIM E(7,2), A(9,2)
020 MAT INPUT E, A
030 FOR K=1 TO 7
040 IF E(K,2) 0 THEN 140
050 LET E(K,2)=-1
060 PRINT K;
070 LET W=E(K,1)
080 IF E(K,1)=0 THEN 30
090 LET T=A(W,2)
100 LET E(T,2)=E(T,2)-1
110 IF A(W,1)=1 THEN 30
120LET W=W+1
130 GO TO 90
140 NEXT K
150 END
```

Program V

```
010 DIM S(9,3)
020 MAT INPUT S
030 FOR L=1 TO 9
040 IF S(L,1)=1 OR S(L,2)=2 THEN 140
050 LET W=S(L,1)
060 LET T=0
070 FOR K=1 TO 9
080 IF S(K,2) W THEN 120
090 LET P=S(K,3)
100 IF P T THEN 120
110 LET T=P
120 NEXT K
130 LET S(L,3)=S(L,3)+T
140 NEXT L
150 LET A=3
160 LET T=0
170 FOR L=1 TO 9
180 IF S(L,2) A THEN 220
190 LET P=S(L,3)
200 IF P T THEN 220
210 LET T=P
220 NEXT L
230 PRINT"EVENT";A;"HAS VALUE";T
240 LET A=A+1
250 IF A=8 THEN 270
260 GO TO 160
270 END
```